

CLAIMS

I claim:

1. A collection makefile generator process for generating makefiles for collections, to be performed on or with the aid of a computer, comprising the following steps:

- (a) receiving a request to generate a makefile for a collection,
(b) accessing collection information for said collection, and
(c) generating a makefile for said collection,

AI
wherein collections are data structures comprised of a collection specifier and collection content containing zero or more collection content files, and wherein a collection specifier contains information about a collection instance, and wherein collection membership information describes collection content,

thereby providing a solution to the collection makefile generator problem, and thereby enabling human programmers to generate collection makefiles in a fully-automated, scalable way that was not previously available.

2. The process of claim 1, wherein

- (a) said step of generating a makefile uses makefile service and makefile fragment template information,

thereby providing a solution to the makefile customization problem, and

thereby enabling human programmers to extensively customize the makefile generation process by controlling the information content of fragments used to implement makefile services, in an automated, scalable way that was not previously available.

output products from a single collection, even where said multiple products must be processed in a particular build order in order to ensure correct results.

8. The process of claim 1, wherein

(a) said step of generating a makefile uses file build order information,

thereby providing a solution to the product file build order problem, and

thereby enabling human programmers to specify a proper build order for multiple product file types within a single collection product, even where the multiple product files must be processed in a particular build order in order to ensure correct results.

9. The process of claim 1, wherein

(a) said step of generating a makefile uses parallelism limit information to calculate and generate parallel makefile targets,

thereby providing a solution to the makefile parallel processing problem, and

thereby increasing the productivity of human programmers by generating makefiles that can be executed in parallel to reduce the time required to perform makefile operations.

10. A programmable collection makefile generator device for generating makefiles for collections, whose actions are directed by software executing a process comprising the following steps:

- AJ
- (a) receiving a request to generate a makefile for a collection,
 - (b) accessing collection information for said collection, and
 - (c) generating a makefile for said collection,

wherein collections are data structures comprised of a collection specifier and collection content containing zero or more collection content files, and wherein a collection specifier contains information about a collection instance, and wherein collection membership information describes collection content,

thereby providing a solution to the collection makefile generator problem, and thereby enabling human programmers to generate collection makefiles in a fully-automated, scalable way that was not previously available.

11. The programmable device of claim 10, wherein

- (a) said step of generating a makefile uses makefile service and makefile fragment template information,

thereby providing a solution to the makefile customization problem, and

thereby enabling human programmers to extensively customize the makefile generation process by controlling the information content of fragments used to implement makefile services, in an automated, scalable way that was not previously available.

12. The programmable device of claim 10, wherein

- (a) said step of generating a makefile uses substitution strings representing

17. The programmable device of claim 10, wherein

(a) said step of generating a makefile uses file build order information,

thereby providing a solution to the product file build order problem, and

thereby enabling human programmers to specify a proper build order for multiple product file types within a single collection product, even where the multiple product files must be processed in a particular build order in order to ensure correct results.

18. The programmable device of claim 10, wherein

(a) said step of generating a makefile uses parallelism limit information to calculate and generate parallel makefile targets,

thereby providing a solution to the makefile parallel processing problem, and

thereby increasing the productivity of human programmers by generating makefiles that can be executed in parallel to reduce the time required to perform makefile operations.

19. A computer readable memory, encoded with data representing a collection makefile generator program that can be used to direct a computer when used by the computer, comprising:

(a) means for receiving a request to generate a makefile for a collection,

(b) means for accessing collection information for said collection, and

A3

(c) means for generating a makefile for said collection,

wherein collections are data structures comprised of a collection specifier and collection content containing zero or more collection content files, and wherein a collection specifier contains information about a collection instance, and wherein collection membership information describes collection content,

thereby providing a solution to the collection makefile generator problem, and thereby enabling human programmers to generate collection makefiles in a fully-automated, scalable way that was not previously available.

20. The computer readable memory of claim 19, wherein

(a) said means for generating a makefile uses makefile service and makefile fragment template information,

thereby providing a solution to the makefile customization problem, and

thereby enabling human programmers to extensively customize the makefile generation process by controlling the information content of fragments used to implement makefile services, in an automated, scalable way that was not previously available.

21. The computer readable memory of claim 19, wherein

(a) said means for generating a makefile uses substitution strings representing collection instance values to replace placeholder strings in makefile fragments,

thereby providing a solution to the multiple product naming problem, and